ОБЕСПЕЧЕНИЕ КАЧЕСТВА ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ ДЛЯ ОБЪЕКТОВ КРИТИЧЕСКОЙ ИНФРАСТРУКТУРЫ

Жарко Е.Ф.

Институт проблем управления им. В.А. Трапезникова РАН,
Россия, г. Москва, ул. Профсоюзная, д.65
zharko@ipu.ru

Аннотация: Разработчики систем для объектов критической инфраструктуры, работая над созданием безопасных систем, придерживаются нормативных документов той отрасли, к которой относятся эти объекты. В этой статье предлагаются практические улучшения в обеспечении качества, которые разработчики систем могут применять в процессе разработки программного обеспечения.

Ключевые слова: программное обеспечение, обеспечение качества, объекты критической инфраструктуры, жизненный цикл.

Введение

Обеспечение качества и управление качеством очень важны при разработке систем, важных для безопасности объектов критической инфраструктуры [1]. Программное обеспечение (ПО) систем для объектов критической инфраструктуры предназначено для решения широкого круга задач и реагирования на возмущения различной природы. Несмотря на принимаемые меры, при работе программного обеспечения таких систем таких систем происходят сбои (как системного программного обеспечения, так и прикладного). Сбои носят систематический характер, так как не всегда возможно проверить всю логику работы программного обеспечения (зачастую это связано с тем, что разрабатываемое программное обеспечение таких систем сильно связано с аппаратным обеспечением и даже небольшие изменения в аппаратном обеспечении могут привести к последующим сбоям). С другой стороны, аппаратные средства ограничены выполнением нескольких задач, поэтому их сбои являются случайными и в основном связаны с деградацией.

Обычно процедура проверки функциональности программного обеспечения заключается в следующем:

- разработчик программного обеспечения проверяет новую функциональность, чтобы убедиться, что модифицированный программный код реализует новые функциональные требования;
- разработчик программного обеспечения проверяет новую функциональность, чтобы убедиться, что программа реализует новые функциональные требования.

Ближе к концу жизненного цикла разработки проводятся заводские приемочные испытания, проводится независимая оценка безопасности, а при необходимости проводится сертификация уполномоченным органом. Результатом процесса является сертификация (заключение) о возможности использования разработанного программного обеспечения для систем объектов критической инфраструктуры.

К объектам критической информационной инфраструктуры относят системы управления технологическими процессами (АСУ ТП) различных объектов критической инфраструктуры, в том числе и атомных электростанций. Эта статья посвящена вопросам обеспечения качества программного обеспечения для систем важных для безопасности АЭС.

1 Обеспечение качества программного обеспечения

На первый взгляд, «качество ПО» может показаться абстрактным понятием. Однако для участников процесса разработки программного обеспечения критерии его качества понятны и измеримы. Сначала рассмотрим общее определение данного понятия: *Качество ПО – комплекс характеристик программного продукта, определяющих способность выполнять возложенные на него функции*.

В настоящий момент этот показатель программного обеспечения регулируется международным стандартом ISO/IEC 25010:2011 [2-4]. Данный стандарт устанавливает многоуровневую систему оценки качества ПО, основанную на восьми базовых характеристиках.

- *Функциональная пригодность* (функциональная полнота, функциональная корректность, функциональная целесообразность). ПО признается функциональным, если выполняет возложенные на него задачи, отвечает заданным потребностям пользователей. Данный аспект предполагает правильную и точную работу, совместимость всех входящих в состав компонентов.

- *Надежность* (завершенность, готовность, отказоустойчивость, восстанавливаемость). Под надежностью ПО понимают бесперебойное выполнение возлагаемых на него задач на заданных условиях в течение установленного времени.
- *Удобство использования* (определимость пригодности, изучаемость, управляемость, защищенность от ошибок пользователя, эстетика пользовательского интерфейса, доступность). Эта характеристика демонстрирует степень удобства ПО для пользователей, его наглядность, легкость эксплуатации и изучения.
- *Уровень производительности* (временные характеристики, использование ресурсов, потенциальные возможности). Характеристике соответствует степень обеспечения продуктом необходимой производительности при заданных условиях.
- *Сопровождаемость* (модульность, возможность многократного использования, анализируемость, модифицируемость, тестируемость). Эта характеристика демонстрирует простоту анализа, тестирования, коррекции компонентов ПО, его обслуживания, а также степень адаптации к новым условиям.
- *Переносимость* (адаптируемость, устанавливаемость, взаимозаменяемость). Степень легкости его переноса на другую платформу. Обеспечение качества ПО предполагает его проверку по каждому из перечисленных параметров, выявление слабых сторон и устранение неисправностей.
- *Совместимость* (сосуществование, интероперабельность). Способность программных компонентов взаимодействовать друг с другом.
- *Защищенность* (конфиденциальность, целостность, неподдельность, отслеживаемость, подлинность), т.е. минимизация угроз, связанных с несанкционированным чтением, изменением информации и т.д. Угрозы могут быть также связаны с некорректным использованием ПО, внешним воздействием со стороны посторонних лиц, выходом из строя технических средств.

Термины «тестирование» и «обеспечение качества», безусловно, связаны, но не тождественны.

Обеспечение качества отвечает за весь процесс разработки и интегрировано на всех его этапах: от создания требований к будущей разработки до тестирования, выпуска релиза продукта и его пострелизное обслуживание.

Задачи специалистов по обеспечению качества ПО включают:

- формирование критериев качества;
- планирование мероприятий для соблюдения критериев качества на каждом этапе разработки;
- выбор средств тестирования;
- тестирование;
- расчет эффективности работы по обеспечению качества;
- предотвращение появления ошибок и усовершенствование процесса.

Тестирование же – это проверка программного обеспечения на соответствие требованиям. Таким образом, что обеспечение качества – это более широкое понятие, которое включает в себя и тестирование. Тестирование может быть автоматизированным или проводиться вручную; может быть полного цикла или направленным на проверку определенного аспекта качества (безопасность, производительность, удобство использования и т.д.).

Тестировщики подготавливают стратегии по тестированию и план на основе спецификации проекта и требований к реализации, создают и оптимизируют набор тестов, осуществляют поиск дефектов, создают и направляют отчеты об обнаруженных дефектах разработчикам, а также проверяют устранение дефектов.

Функция обеспечения качества может выполняться внутренним отделом компании, а может делегироваться независимому подрядчику [5, 6], который объективно оценит само решение, усовершенствует процессы обеспечения качества и тем самым позволит выпустить на рынок продукт более высокого качества, отвечающий бизнес-требованиям и ожиданиям пользователей.

2 Подтверждение качества программного обеспечения

Программное обеспечение систем, важных для безопасности АЭС в части жизненного цикла систем данного типа, обычно соответствует V-модели [5, 6], в которой предусмотрены этапы планирования, разработки технических требований и спецификации архитектуры, рабочий проект и реализация. И на каждом этапе составляется соответствующий отчет о верификации для правой части модели, проводятся верификация и валидация. После этапа реализации начинается процесс проверки

снизу вверх вплоть до установки и ввода в эксплуатацию, а также возможного сопровождения и вывода из эксплуатации (см. рис. 1). V-модель не является строго соблюдаемой для гибких методов разработки, таких как Scrum [7-9] и Kanban [10, 11], в отличие от некоторых методов [12]. В V-модели может присутствовать элемент итеративной разработки, если деятельность по разработке тщательно спланирована. Создание надежной, безопасной и отказоустойчивой системы требует наличия четко определенных стандартов, процессов и соглашений, которые легко применять и проверять.

Некоторые из ограничений, накладываемых на языки программирования, используемые для разработки ПО систем, важных для безопасности АЭС, включают: строгую типизацию, поддержку безопасного и структурированного программирования. Тот факт, что язык программирования удовлетворяет этим ограничениям, не означает, что все операции на этом языке программирования безопасны. Чтобы устранить недостатки выбранного языка программирования, обычно используют только подмножество языка программы, отвечающее наложенным ограничениям. Стандарты и соглашения программирования используются для обеспечения подмножества языка программирования и предотвращения небезопасных операций кодирования разработчиками ПО. При разработке программного обеспечения необходимо использовать стандарты программирования [5, 6]. После согласования соглашения о кодировании разработчики ПО должны поддерживать его целостность. Любое отклонение от правил программирования должно быть обоснованы.

Разработчики ПО могут выделять многочисленные сценарии поведения ПО из требований и преобразовывать эти сценарии в алгоритмы. Но не менее, сложность системы может привести к сложным алгоритмам, которые очень сложно поддерживать и при необходимости модифицировать. Обзор программного кода является одним из методов, используемых для упрощения и сопровождения программного кода. По сути, программная архитектура систем, важных для безопасности, должна быть модульной, слабо связанной и согласованной. Все изменения и разработка исходного программного кода должны быть проверены и приняты, чтобы программный код мог быть зарегистрирован в репозитории.

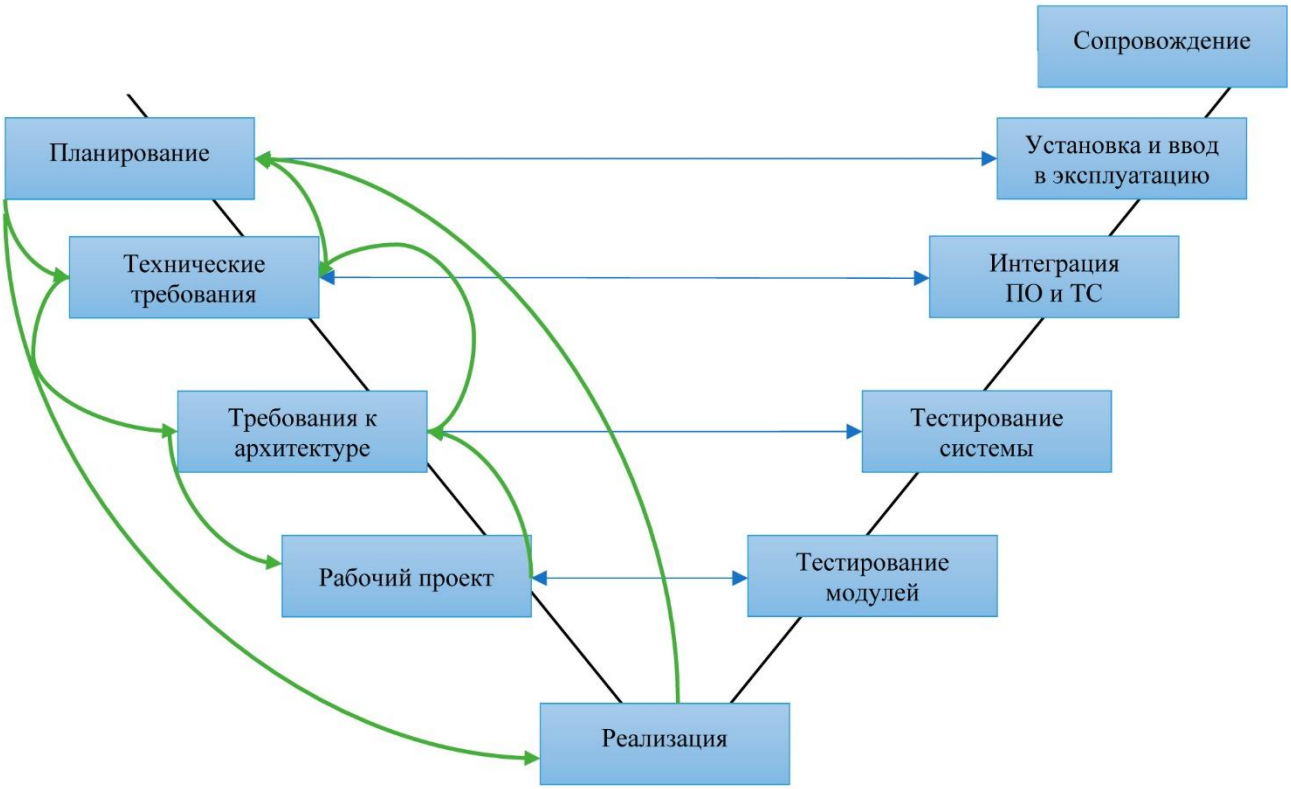

Рис. 1. Итеративная V-модель

Управление конфигурацией является неотъемлемой частью общей системы обеспечения качества ПО. Должны быть разработаны правила, регулирующие все изменения программного обеспечения в течение жизненного цикла проекта. Ревизии всех документов и программного кода должно корректно отслеживаться с помощью специальной базы данных. Запросы на изменение и отчеты о

несоответствиях документируются и отслеживаются в базе данных от анализа до проверки. План обеспечения защищенности и план управления конфигурацией обычно должны включать несколько этапов обеспечения качества программного обеспечения. Эти этапы разработки проходят валидацию, и предоставляется новая версия программного обеспечения для тестирования подсистем, интеграции и тестирования системы в целом.

Программы, какими бы стабильными они не были, могут потребовать модификации. В течение жизненного цикла проекта заказчик может сделать запрос на изменение с новыми требованиями, а отчет о несоответствии может быть представлен для анализа и реализации. В результате, это может привести к модификации программного обеспечения и последующей его поставке для установки и ввода в эксплуатацию. В команде разработчиков, в которой разные программисты модифицируют разные модули / компоненты и подсистемы, в процессе тестирования могут возникать ошибки. Одной из методологий обеспечения качества программного обеспечения для поддержания целостности исходного кода является автоматизированный процесс непрерывной интеграции.

В каждом проекте есть процесс управления требованиями, который тщательно контролируется на протяжении всего жизненного цикла проекта. Требования заказчиков, включая законодательные и нормативные требования, связанные с уровнем защищенности, регистрируются как базовые перед началом каждого детального проектирования и реализации. Такая практика обеспечения качества программного обеспечения гарантирует, что требования можно отслеживать от детального проектирования и относятся это распределение требований и матрица прослеживаемости требований [13]. Распределение требований очень хорошо подходит для реализации функциональных требований.

В идеале жизненный цикл разработки ПО при бескомпромиссном применение вышеупомянутых передовых практик приведет к созданию безопасного и качественного ПО. Программное обеспечение, с другой стороны, не осязаемо; оно состоит из множества сложных алгоритмов с множеством путей выполнения и результатов. Исходя из многолетнего опыта, процедуры оценки систем, важных для безопасности АЭС, не являются достаточно надежными, чтобы гарантировать безопасность и отказоустойчивость системы. Дело в том, что большинство процедур, используемых для независимой оценки безопасности ПО для систем, важных для безопасности АЭС, являются скорее теоретическими, чем эмпирическими. Это связано с тем, по большей части, проверяется только заявление поставщика и валидатора ПО о том, что процессы жизненного цикла проекта или продукта выполняются. То есть это процесс включает в себя проверку того, что указанные процессы соответствуют юридическим и нормативным обязательствам.

Таким образом, оценка качества не является гарантией безопасности использования подсистемы для управления, мониторинга и диагностики. Несмотря на это отсутствие уверенности, оценка качества ПО используется для систем, важных для безопасности АЭС, как свидетельство уверенности в том, что система соответствует своему назначению.

3 Методология оценки качества программного обеспечения

Организация оценки качества программных систем и ее отличительные черты тесно связаны со спецификой разработки программного обеспечения для систем, важных для безопасности АЭС [14, 15]. Отличительными чертами такой оценки качества являются, во-первых, значительная повторяемость (многократность) оценки, во-вторых, приближенный характер единичной оценки, в-третьих, не стремление сделать окончательный вывод о качестве программного обеспечения, а выявление тенденции улучшения качества ПО, а также отдельных аномалий. Поэтому, именно итерация разработки как всего ПО, так и его отдельных модулей, определяет степень рекурсивности методологии оценки качества программного обеспечения для систем важных для безопасности АЭС.

Методы подбора экспертов, создания экспертных групп и получения данных от экспертов для оценки как показателей важности $_{(\ldots)}$, так и показателей качества программного обеспечения $\{(q_i)\}$ считаются внешними по отношению к предлагаемой методологии.

В рамках методологии считается, что все итерации оценки качества программного обеспечения проводятся с использованием единого математического представления интегрального критерия качества ПО. Сопоставление профилей качества, вычисленных на основе различных форм свертки показателей качества более низкого уровня при вложении более сложного представления интегрального критерия качества, не разрешается. Модификация используемого математического представления интегрального критерия качества приводит к «перезапуску» процесса накопления статистики по итерациям оценки качества программного обеспечения.

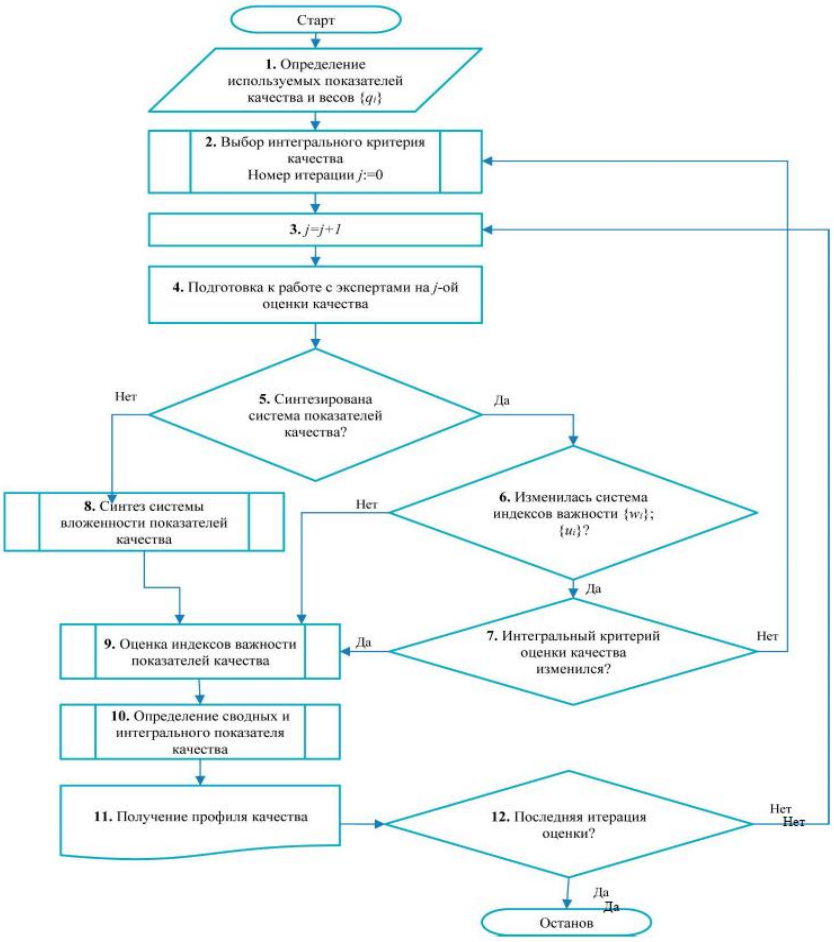

Рис. 2. Обобщенный алгоритм оценки качества программного обеспечения

Методология предполагает ее дальнейшую автоматизацию и реализацию в виде соответствующего ПО в гибких средах разработки программного обеспечения.

Алгоритм оценки качества программного обеспечения (см. рис. 2) можно представить в виде 12 этапов вычислительного процесса.

Предполагается дальнейшая детализация предложенного алгоритма методики оценки качества программного обеспечения, поскольку на рис. 2 алгоритм представлен на высоком уровне логического обобщения. Детальное описание алгоритма определяется узкоспециализированными вопросами назначения систем, для которых разрабатывается ПО. В то же время данный алгоритм не предполагает установления предельного количества итераций квалиметрической оценки качества программного обеспечения. Предлагаемая методология оценки качества программного обеспечения не является самостоятельной и может использоваться как элемент применительно к технологической схеме разработки программного обеспечения.

Представленный на рис.2 алгоритм оценки качества программного обеспечения с дальнейшим его уточнением применительно к определенным условиям применяемой технологической системы разработки программно-технического комплекса может быть оптимизирован по временным затратам количеству полезных итераций оценки и других атрибутов организационного и технического характера.

Параметр i, указанный в алгоритме, является номером итерации, а $\{\}$ являются обобщенными индексами важности. Это число, часто, соответствует номеру временного интервала в технологической системе разработки или количеству ежедневных итераций разработки, после которых оценивается качество разработанного ПО. Стоит отметить, что принятие решения о модификации используемого математического представления интегрального критерия оценки качества ПО приводит к «автоматическому» обнулению количества рассматриваемых итераций оценки ($i \_ 0$) и, соответственно, к перезапуску анализа качества программного обеспечения в соответствии с новой принятой формой критерия качества.

Заключение

Обобщенный алгоритм оценки качества ПО систем, важных для безопасности, связывает воедино логическую последовательность использования необходимых процедур оценки качества, а также описывает логику формирования циклического вычислительного процесса при оценке качества в рамках технологической системы разработки ПО. Также учитывается улучшение некоторой ограниченной функциональности, выбранной в рамках одного списка задач подсистемы, который полностью охватывает ее функциональность, реализованную в проекте, для разработки и улучшения качества ПО.

Предлагаемая методология оценки качества может быть применена в более широкой области использования, а также и может находить применение в различных технологических системах для разработки программного обеспечения. В настоящее время она используется для обеспечения качества программного обеспечения систем верхнего уровня АСУ ТП АЭС.

Литература

1. *Bullock J.A., Haddow G.D., Coppola D.P.* Introduction to Homeland Security. Chapter 8 “Cybersecurity and critical infrastructure protection”. – Elsevier, 2021. – P.425-497.
2. *ГОСТ Р ИСО/МЭК 25010-2015.* Системная и программная инженерия - Требования и оценка качества систем и программного обеспечения (SQuaRE) -Модели качества систем и программных продуктов.
3. *Jharko, E.Ph.* Evaluation of the Quality of a Program Code for High Operation Risk Plants // IFAC Proceedings Volumes. Vol. 47. 2014, iss. 3. – P.8060-8065.
4. *Jharko E.* Towards the quality evaluation of software of control systems of nuclear power plants: Theoretical grounds, main trends and problems // Proceedings of the 12th International Conference on Informatics in Control, Automation and Robotics. Colmar, France. 2015. – P.471-478.
5. *IEC 60880:2006.* Nuclear power plants - Instrumentation and control systems important to safety - Software aspects for computer-based systems performing category A functions.
6. *IEC 62138:2018.* Nuclear power plants - Instrumentation and control systems important to safety - Software aspects for computer-based systems performing category B or C functions.
7. *Santos Júnior P.S., Barcellos M.P., Falbo R. de A., Almeida, J.P.A.* From a Scrum Reference Ontology to the Integration of Applications for Data-Driven Software Development // Information and Software Technology. Vol. 136. 2021, 106570.
8. *Prasetya K.D., Suharjito, Pratama, D.* Effectiveness Analysis of Distributed Scrum Model Compared to Waterfall approach in Third-Party Application Development // Procedia Computer Science. Vol. 179. 2021. – P.103–111.
9. *Morandini M., Coleti T.A., Oliveira E., Corrêa, P.L.P.* Considerations about the efficiency and sufficiency of the utilization of the Scrum methodology: A survey for analyzing results for development teams // Computer Science Review. Vol. 39. 2021, 100314.
10. *Ahmad M.O., Dennehy D., Conboy K., Oivo M.* Kanban in software engineering: A systematic mapping study // Journal of Systems and Software. Vol.137. 2018. – P.96–113.
11. *Lei H., Ganjeizadeh F., Jayachandran P.K., Ozcan P.* A statistical analysis of the effects of Scrum and Kanban on software development projects // Robotics and Computer-Integrated Manufacturing. Vol. 43. 2017.– P.59–67.
12. *Biesialska K., Franch X., Muntés-Mulero V.* Big Data analytics in Agile software development: A systematic mapping study // Information and Software Technology. Vol. 132. 2021, 106448.
13. *Hill J., Tilley S.* Creating safety requirements traceability for assuring and recertifying legacy safety-critical systems // Proceedings of the 18th IEEE International Requirements Engineering Conference. 2010. – P.297-302.
14. *Jharko E.* Formalizing the Safety Functions to Assure the Software Quality of NPP Safety Important Systems // Proceedings of the 16th International Conference on Informatics in Control, Automation and Robotics (ICINCO 2019, Prague, Czech Republic). Vol. 2. 2019. – P.637-644.
15. *Jharko E.* Systems Important for NPP Safety: Software Verification and Cybersecurity // Lecture Notes in Electrical Engineering. Vol. 729. 2021. – P.90-98.